Introduction to Pleiades

Johnny Chang
NAS Division
NASA Ames Research Center

2012 Summer Short Course for Earth System Modeling and Supercomputing

Outline

- Computing resources available at NAS
- Logging in to Pleiades
- Transferring files to/from Pleiades
- Setting up your module environment
- Compiling your code
- Running jobs with PBS
- Working with PBS
- Lustre Best Practices

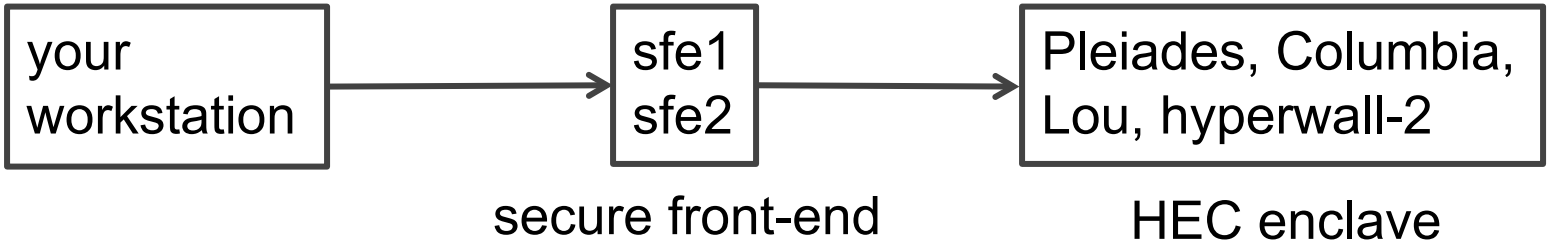
NAS Systems

- **Pleiades: 11,776-node Intel Xeon cluster**
  processor family: x86_64
  - 4096 Harpertown nodes: 8 cores and 8GB per node
  - 1280 Nehalem nodes: 8 cores and 24GB per node
  - 4672 Westmere nodes: 12 cores and 24GB per node
  - 1728 Sandy Bridge nodes: 16 cores and 32GB per node
- **Columbia: 4 large Single-System-Image systems**
  processor family: ia64
  - Columbia21: 512 CPUs and 1 TB memory
  - Columbia22: 2048 CPUs and 4 TB memory
  - Columbia[23,24]: 1024 CPUs and 2 TB memory each
- **Lou: 14 PB mass storage system**
  processor family: ia64

Pleiades front-ends

- **pfe1, pfe2, …, pfe12**
  - Harpertown nodes: 8 cores,16 GB/node, 1GigE network
  - Used for logging in, and interactive work: editing, compiling, submitting jobs, etc.
- **bridge1, bridge2**
  - Harpertown nodes: 8 cores, 64 GB/node, 10 GigE network
  - Larger memory for pre- or post-processing, viewing graphics (matlab, tecplot, idl, etc.)
  - Better network for transferring large files (especially to Lou)
- **bridge3, bridge4**
  - Nehalem-EX nodes: 32 cores, 256 GB/node, 10 GigE network
  - Newer, larger bridge nodes

Logging in to NAS systems



Two-step connection method: Easy, but not recommended

- First, from your wks, login to the secure front-end

  your_wks% ssh sfe1.nas.nasa.gov (or sfe2.nas.nasa.gov)

  or

  your_wks% ssh username@sfe1.nas.nasa.gov

  (if your NAS username is different)

  This step requires 8-char pin + passcode from fob and password (two-factor authentication)

- Second, from sfe1 (or sfe2), login to Pleiades front-end, pfe

  sfe1% ssh pfe

  This step requires password

Logging in to NAS systems

One-step connection method: preferred method

your_wks% ssh pfe

Enter 8-char pin + passcode from fob

This requires setting up SSH Passthrough

("pass through" because no direct login to sfe1 or sfe2)

Setting up SSH Passthrough

1. **On your workstation:**
   - ssh-keygen -t rsa (choose a passphrase, this command will generate two files: id_rsa and id_rsa.pub)
   - Copy public key to sfe1 (and/or sfe2)
     scp id_rsa.pub username@sfe1.nas.nasa.gov:~/.ssh2
2. **On sfe1 (and/or sfe2):**
   - echo "Key id_rsa.pub" > ~/.ssh2/authorization
   - Copy public key to pfe and lou
3. **On pfe and lou:**
   - mv id_rsa.pub ~/.ssh/authorized_keys
4. **On your workstation:**
   - Download the config file from hecc webpage on SSH Passthrough, edit and enter your username, and save the config file under your ~/.ssh directory
   - Start ssh-agent
     eval `ssh-agent`
     ssh-add (type your passphrase when prompted)

File Systems

- **$HOME file system is NFS**
  - disk quota: 8GB soft and 10GB hard limit
  - 14 days grace period over soft quota
  - files backed up everyday
  - check quota with: quota -v

- **Scratch directory: /nobackup/userid is a Lustre file system**
  - disk quota: 500GB soft and 1TB hard limit
  - inode quota: 75000 soft and 100000 hard limit
  - 14 days grace period over soft quota
  - files and directories are never backed up
  - check quota with: lfs quota -u userid /nobackupp[1-6] (use your nobackupp number)

Transferring files to/from NAS systems

Easy if SSH passthrough is already set up

Examples:

- wks% scp file1 pfe: transfer file1 to pfe
- wks% scp file1 pfe:file2 transfer and rename to file2 on pfe
- wks% scp file1 pfe:dir1 transfer to dir1 on pfe
- wks% scp -r dir1 pfe: recursively copy dir1 and its contents to pfe
- wks% scp pfe:path_to/file1 . transfer file1 from pfe to your workstation

Only requires pin + passcode

Use Secure Unattended Proxy to avoid pin+passcode

More cumbersome if SSH passthrough is not set up

- Need to transfer twice, either through sfe[1,2] (not recommended, limited disk space) or through dmzfs[1,2]
- File transfer cannot be initiated from dmzfs1/dmzfs2 because of their "jailed" environments (limited Unix commands and non-functional ssh or scp commands). Files can be "pushed" into or "pulled" out of dmzfs[1,2]
- Files are automatically deleted from dmzfs[1,2] after 24 hours

Setting up module environment

New account created with no default compilers
(except for GNU compilers)

module avail shows all 172+ modules available
(36 compilers, 22 MPI libraries, 12 HDF5 libraries, etc.)

Recommend adding the following to the end of your .login file:

module load comp-intel/2012.0.032 mpi-sgi/mpt.2.06a67

(don't load MKL modules, it's already included in v.11 or later Intel compiler modules)

Default shell is csh (same as tcsh)

Contact control-room if you want a different default shell

Useful module commands:

- module list (list currently loaded modules)
- module purge (unloads all currently loaded modules)
- module switch current_module new_module
- module show some_module (shows how your environment variables, PATH, FPATH, LD_LIBRARY_PATH, etc. are changed by loading the module)
- module help some_module (info on how some_module was built)

Compiling and Building your code

Intel compilers: ifort – Fortran compiler
icc – C compiler
icpc – C++ compiler

Compiler options:

aggressive optimization:	-O3 -ip
maintain precision:	-fp-model precise (lowers optimization)
large arrays > 2GB:	-mcmodel=medium
	-shared-intel (needed at link step)
debugging:	-g -traceback -fpe0 -check

Linking:

MKL math library:	-mkl=sequential
SGI's MPI library:	-lmpi

Example:

```
ifort -c -O3 -ip file1.f90
ifort -c -O3 -ip file2.f90
ifort -o my_exec file1.o file2.o -lmpi
```

Running jobs with PBS

Sample PBS script (run.scr):

```
#PBS -l select=16:ncpus=8:model=har
#PBS -l walltime=1:00:00
#PBS -j oe
cd $PBS_O_WORKDIR
mpiexec -np 128 ./my_exec  >  output
```

% qsub run.scr (submit PBS job)

227697.pbspl1.nas.nasa.gov

qstat -au jsmith	(shows all jobs running or queued by user jsmith)
qstat -su jsmith	(gives a one line explanation for status of jsmith's jobs)
qstat -nu jsmith	(shows nodes used by jsmith's running jobs)
qstat -r	(shows all running jobs)
qstat -i	(shows all queued jobs sorted by priority)
qdel 227697	(delete job 227697)

Running jobs with PBS (continued)

- **'devel' queue for faster turnaround (Westmere and Sandy Bridge nodes only)**
  - Each user can run only one job at a time in the devel queue for up to 2 hours
  - Submit jobs with: qsub -q devel@pbspl3 run.scr
    12709.pbspl3.nas.nasa.gov
  - qstat -r devel@pbspl3 (shows all running jobs in the devel queue)
  - qstat -i @pbspl3 (shows all queued jobs served by pbspl3)
- **Interactive PBS jobs (qsub -I)**
  - qsub -I -lselect=4:ncpus=12:model=wes,walltime=5:00:00
  - qsub -I -q devel@pbspl3 -lselect=4:ncpus=12:model=wes,walltime=2:00:00
  - qsub -I -v DISPLAY -lselect=4:ncpus=8:model=neh
    qsub: waiting for job 227786.pbspl1.nas.nasa.gov to start
    (Ctrl-c if you don't want to wait)
  - Default is 1 hour if you don't specify walltime
  - More predictable start time running interactive PBS job in devel queue

Lustre Best Practices

Pleiades scratch directory, /nobackup/jsmith, is a Lustre filesystem

/nobackup/jsmith is a symlink to the actual directory: pfeX% ls -l /nobackup/jsmith

lrwxrwxrwx 1 root root 18 Jul 15 16:53 /nobackup/jsmith -> /nobackupp1/jsmith/

Checking quotas on Lustre

% lfs quota -u jsmith /nobackupp1

Disk quotas for user jsmith (uid xxxx):

Filesystem	kbytes	quota	limit	grace	files	quota	limit	grace
/nobackupp1	97757456	210000000	420000000	-	42573	75000	100000	

File striping (necessary if file is greater than 1GB or read by many procs)

lfs setstripe -c 16 -s 4m bigfile (Sets stripe count of 4 and stripe size of 4MB for bigfile; must be done before bigfile is created)

lfs gestripe bigfile (get information on file striping for bigfile)

lfs setstripe -c 16 -s 4m bigdir (sets striping for directory bigdir; all new files created under bigdir will retain the file striping characteristics of bigdir)

Default file striping is -c 1 -s 4m

Lustre Best Practices (cont.)

Avoid repetitive or continuous file *stats* by adding *sleep*

For example, if checking for the presence of file "GO," instead of:

```
while (! -e GO)
end
```

use

```
while(! -e GO)
sleep 2
end
```

For more on Lustre Best Practices, go to http://www.nas.nasa.gov/hecc and search for Lustre. Start with "Lustre Basics."